Otimização de Processos (COQ897)

Prof. Argimiro R. Secchi

Terceira Lista de Exercícios - 2020

José Rodrigues Torraca Neto

8 (b) Fizemos uma transformação em uma das colunas do dataframe que cada método produz, que será explicada aqui.

Primeiro, temos que explicar um conceito importante, o de arrays estruturados:

Em ***numpy***, arrays estruturados são ndarrays cujo tipo de dados é uma composição de tipos de dados mais simples organizados como uma sequência de campos nomeados.

Por exemplo:

```
import numpy as np

x = np.array([('Rex', 9, 81.0), ('Fido', 3, 27.0)],
       dtype=[('name', 'U10'), ('age', 'i4'), ('weight', 'f4')])
x
```

```
array([('Rex', 9, 81.), ('Fido', 3, 27.)],
      dtype=[('name', '<U10'), ('age', '<i4'), ('weight', '<f4')])
```

Aqui, x é uma matriz unidimensional de comprimento dois cujo tipo de dados é uma estrutura com três campos:

1. uma string de comprimento 10 ou menor chamada 'name',
2. um inteiro de 32 bits chamado 'age' e
3. um float de 32 bits denominado 'weight'.

Se você indexar x na posição 1, obterá uma estrutura:

```
x[1]
```

```
('Fido', 3, 27.)
```

Tipos de dados estruturados são projetados para serem capazes de imitar "estruturas" na linguagem C e compartilhar um layout de memória semelhante. Eles se destinam à interface com o código C e à manipulação de baixo nível de buffers estruturados, por exemplo, para interpretar blobs binários.

No entanto, especificamente no caso da função que definimos para o plot das curvas de níveis, queríamos que o input de X fosse uma simples ndarray simples do numpy:

```
# Função para criar um plot de curvas de níveis:

def contour(sympy_function):
    x = np.linspace(150, 300, 100)
    y = np.linspace(350, 500, 100)
    x, y = np.meshgrid(x, y)
    func = f_x(sympy_function, np.array([x,y]))
    return plt.contour(x, y, func)

# Função para plotar com o caminho do algoritmo:

def contour_travel(x_array, sympy_function):
    x = np.linspace(150, 300, 100)
    y = np.linspace(350, 500, 100)
    x, y = np.meshgrid(x, y)
    func = f_x(sympy_function, np.array([x,y]))
    plt.contour(x, y, func)
    plot = plt.plot(x_array[:,0],x_array[:,1],'x-')        # Aqui entra a ndarray de x[x1,x2];
    return (plot)                                          # já sequenciada de 0 a n (indicado pelo símbolo ":").
```

Analisando o final de um dos métodos que implementamos:

```
print('Nº de steps que o método Steepest Descent levou para convergir: ', len(points_checked))

print("Tabela com as iterações: ")
points_checked

print("caminho do método Steepest Descent sobre as curvas de níveis:")
contour = contour(f)

y = points_checked['xk'].values.astype(dtype=[('f0', '<f8'), ('f1', '<f8')])    # transformação

y

Contour_path = contour_travel(y.view(np.float64).reshape(y.shape + (-1,)), f)
```

Se analisarmos a coluna 'xk' do dataframe points_checked, antes da transformação, teríamos:

```
points_checked['xk'].values
```

```
    array([(500, 500), (250.01, 593.78), (231.14, 543.47), (237.86, 503.83),
           (222.07, 499.76), (226.29, 483.42), (218.68, 481.45),
           (220.72, 473.54), (217.02, 472.59), (218.02, 468.74),
           (216.21, 468.27), (216.7, 466.39), (215.82, 466.17),
           (216.06, 465.25), (215.63, 465.14), (215.74, 464.69),
           (215.53, 464.64), (215.59, 464.42), (215.49, 464.39),
           (215.52, 464.29), (215.46, 464.27), (215.48, 464.22),
           (215.45, 464.21), (215.46, 464.19), (215.45, 464.19),
           (215.45, 464.17), (215.45, 464.17), (215.45, 464.17),
           (215.44, 464.17), (215.45, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
           (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
```

Esta é uma array estruturada, com dtype='object'. É uma array que o panda gera, ao usarmos o comando usual ".append" de adicionar um item a uma coluna do dataframe a cada iteração, no caso a coluna "xk".

Se usarmos essa array, o python reconhecerá como uma array unidimensional, e não bidimensional como queríamos para o input da função countour_travel.

E pelo fato dela ser dtype='object', não podemos fazer certas transformações com esta array.

Então, temos que primeiro transformar essa array com o comando ".astype" em uma array bidimensional de floats:

```
points_checked['xk'].values.astype(dtype=[('f0', '<f8'), ('f1', '<f8')])
```

```
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16), (215.44, 464.16),
       (215.44, 464.16), (215.44, 464.16)],
      dtype=[('f0', '<f8'), ('f1', '<f8')])
```

Finalmente, agora podemos transformar a nossa array.

O comando np.reshape dá uma nova forma a uma array sem alterar seus dados, no caso de apenas uma coluna.

E o comando np.view permite que transformemos os elementos da array para outro tipo de objeto, no caso np.float64 bits, que é alocado na memória:

```
y = points_checked['xk'].values.astype(dtype=[('f0', '<f8'), ('f1', '<f8')])

y
```

```
y.view(np.float64).reshape(y.shape + (-1,))
```

```
array([[500.  , 500.  ],
       [250.01, 593.78],
       [231.14, 543.47],
       [237.86, 503.83],
       [222.07, 499.76],
       [226.29, 483.42],
       [218.68, 481.45],
       [220.72, 473.54],
       [217.02, 472.59],
       [218.02, 468.74],
       [216.21, 468.27],
       [216.7 , 466.39],
       [215.82, 466.17],
       [216.06, 465.25],
       [215.63, 465.14],
       [215.74, 464.69],
       [215.53, 464.64],
       [215.59, 464.42],
       [215.49, 464.39],
       [215.52, 464.29],
       [215.46, 464.27],
       [215.48, 464.22],
       [215.45, 464.21],
       [215.46, 464.19],
       [215.45, 464.19],
       [215.45, 464.17],
       [215.45, 464.17],
       [215.45, 464.17],
       [215.44, 464.17],
       [215.45, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
       [215.44, 464.16],
```